УДК 004.4

Юстименко Є. А., здобувач 3 курсу спеціальності 122 Комп'ютерні науки,
Труханська В. О., здобувачка 3 курсу спеціальності 122 Комп'ютерні науки,
Ніколюк П. К., д-р фіз.-мат. наук, професор, професор кафедри інформаційних технологій

ПАТЕРНИ ПРОЄКТУВАННЯ У ПРОГРАМУВАННІ

Донецький національний університет імені Василя Стуса, м. Вінниця

Патерни проєктування – рішення для певних проблем, які часто зустрічаються під час проєктування архітектури програмного забезпечення. Різниця патернів та готових бібліотек у тому, що патерн не є конкретним кодом чи рішенням, а лише загальним принципом для вирішення проблеми, яка виникає під час проЄктування програми. Патерн не можна скопіювати, його завжди потрібно реалізовувати під конкретну архітектуру ПЗ [1].

Плюсами використання патернів є:

1. Стандартизація коду – патерни мають готові рішення, які можна просто реалізовувати у своїх проєктах, не витрачаючи час на проєктування власної архітектури.

2. Типові рішення – більшість патернів перевірені та оптимізовані, що дає змогу уникнути зайвих проблем з архітектурою проєкту.

3. Зміни у структурі – патерни спрощують внесення змін до проєкту без значного впливу на всю програму.

4. Сприйняття структури – патерни значно покращують читабельність коду проєкту, що дає змогу іншим розробникам швидше орієнтуватися у проєкті, його функціях та архітектурі.

Розглянемо основні патерни, які найчастіше використовуються в сучасному програмному забезпеченні.

Одинак (Singelton) – патерн проєктування, який передбачає, що один клас має один екземпляр та одну точку вводу у програмі відповідно. Тобто патерн призначений для того, щоб гарантувати, що клас має тільки один екземпляр і забезпечує глобальний доступ до нього. Використання патерну сприяє поліпшенню читабельності коду, оскільки він визначає єдину точку доступу до об'єкта, уникнення дублювання та спрощення механізмів управління ресурсами.

Прототип (Prototype) – патерн проєктування, який призначений для копіювання об'єктів без розгляду їх внутрішньої структури та реалізації, шляхом створення нового екземпляру класу методами, реалізованими в цьому ж класі.

Фабричний метод (Factory) – патерн проєктування, який визначає загальний тип та ознаки об'єкта, реалізованого класом, і дає змогу визначати різні

значення цих ознак для його підкласів. Цей метод сприяє легкості розширення програмних систем, оскільки дає змогу додавати нові класи без зміни коду вже наявних класів.

Абстрактна фабрика (Abstraction factory) – патерн проєктування, що дає змогу створювати групи пов'язаних об'єктів, не прив'язуючись до конкретних класів створюваних об'єктів.

Будівельник (Builder) – патерн проєктування, що дає можливість покроково створювати складні об'єкти, використовуючи один код [2].

Інновації у сфері патернів проєктування можуть бути спрямовані на покращення ефективності, гнучкості та адаптивності в програмному забезпеченні. Однією з інновацій може стати розробка динамічних патернів проєктування. Основна ідея полягає в тому, щоб розширювати та змінювати патерни в реальному часі, щоб забезпечити адаптивність системи до динамічних вимог. Можливі переваги динамічних патернів.

1. Автоматичне адаптування (система може самостійно визначати, коли необхідно застосовувати той чи інший патерн, в залежності від поточного стану або обставин).

2. Автоматичне управління ресурсами (система може динамічно вирішувати, коли застосовувати ефективні патерни для оптимізації використання ресурсів).

3. Динамічні зміни у реальному часі (можливість миттєвої зміни патернів без перезапуску програмного забезпечення, що дає змогу більш швидко реагувати на змінні умови).

4. Самоналаштування і адаптація (можливість системи вивчати та адаптуватися до оптимальних патернів на основі досвіду та даних використання у залежності від умов у системі).

5. Зменшення кількості коду (динамічне застосування патернів може призвести до меншого обсягу коду та більшої гнучкості).

Згадані патерни дають змогу підвищити адаптивність системи до змін та зробити розробку програмного забезпечення більш гнучкою та ефективною. Важливо, щоб динамічні зміни патернів були добре збалансованими та мали чіткі стратегії управління, щоб уникнути складнощів у розумінні й обслуговуванні системи [3].

Список використаних джерел

1. Що таке патерни проєктування. URL: https://robotdreams.cc/uk/blog/301-chto-takoe-patterny-proektirovaniya (дата звернення: 12.11.2023).

2. Огляд патернів проєктування. URL: https://www.globallogic.com/ua/insights/blogs/dive-into-the-selenium-patterns/ (дата звернення: 13.11.2023).

3. Design Patterns in the Teaching of Programming. URL: https://www.sciencedirect.com/science/article/pii/S1877042814044218 (дата звернення: 12.11.2023).